# 3. OVERFLOW-Mode Operation

## 3.1 Required Input Files

OVERFLOW 2.2 can be run for single grid or multiple grid steady and unsteady flow applications without using the OVERFLOW-D options in the code. This is referred to as OVERFLOW-mode, as opposed to OVERFLOW-D-mode. For single grid applications that do not require loads calculations, only two input files are required to run the code: the grid file (**grid.in**) and a NAMELIST input file (**over.namelist**). The format of the grid file is given in Appendix A. The grid must be a FORTRAN unformatted PLOT3D file. Single- or multiple-grid PLOT3D format may be used for single grids. The grid must match the precision (single- or double-precision) and the ENDIAN-ness (big- or little-ENDIAN) that was selected when the flow solver was compiled. Utility codes are provided to convert the files if they are not in the proper format for the flow solver. Grids should be checked to assure that they are right handed and have no negative volumes. These checks can be performed by running OVERFLOW for 0-steps (which also checks the input files for errors), or by using the **overgrid** graphical interface in Chimera Grid Tools. The full NAMELIST input is given in Appendix B.

For 2D grids, three parallel planes must be supplied. The grid planes must lie in the $(x,z)$ plane, offset by a constant distance in $y$. Any constant spacing between the planes may be used, but $y$ values of 1, 0, and -1 are conventional. For computational efficiency, 2D problems should be oriented to use 3 JK-planes, rather than JL- or KL-planes.

For axisymmetric problems, again a three-plane grid must be supplied. The center plane must lie on the $(x,z)$ plane (i.e., at $y$=0). The axis of symmetry is the $x$-axis, and the first and third planes must lie at -/+1° (and not +/-1°) rotations from this plane. Again, the $(j,k,l)$ system must be right-handed and the grid should be oriented to use 3 JK-planes, rather than JL- or KL-planes for computational efficiency.

OVERFLOW 2.2 assumes that all grid blocks communicate using overset interpolation. The interpolation can be set up using the grid assembly code contained in OVERFLOW 2.2 when the code is run in OVERFLOW-D-mode. An additional input file is required when not running in OVERFLOW-D-mode. The file may be generated using one of the grid assembly codes PEGASUS 5[1] or SUGGAR[2]. This grid assembly file is named **XINTOUT**. The **XINTOUT** file format is specified in Appendix A. This file must also match the precision (single- or double-precision) and the endian-ness (big- or little-endian) that was selected when the flow solver was compiled.

## 3.2 Force/Moment Integration

A positive value of the **&GLOBAL** NAMELIST input **NFOMO** will calculate and report force and moment coefficients every **NFOMO** iterations. These force and moment calculations use the overset grid FOMOCO[3] or USURP[4] utilities. The user is required to set up an input file and to run **mixsur** or **usurp** before the OVERFLOW 2.2 flow solution proceeds. Based on the input surfaces specified, **mixsur** or **usurp** determines a unique integration surface with no overlap. OVERFLOW 2.2 requires some of the files created by **mixsur** (**mixsur.fmp**, **grid.ibi**, and **grid.ptv**) or **usurp** (**mixsur.fmp**, **grid.ibi**, and **panel_weights.dat**) in order to perform the force/moment calculation. It is the user's responsibility to check the **mixsur** or **usurp** output to confirm that suitable integration surfaces were created. Force and moment coefficients are written to **fomoco.out**. The format of the **fomoco.out** file is shown in Appendix A.

For typical CFD calculations, the body axes are chosen such that the $x$-axis points nose-to-tail, $z$ points up, and $y$ is out the right wing. The angle-of-attack ($\alpha$) and sideslip ($\beta$) angles are then defined as

$$\begin{aligned} \alpha &= \tan^{-1}\left(\frac{w_\infty}{u_\infty}\right) \\ \beta &= \sin^{-1}\left(\frac{-v_\infty}{V_\infty}\right) \end{aligned} \tag{3.1}$$

where $V_\infty$ is the magnitude of the free-stream wind, which becomes $(u,v,w)_\infty$ when resolved in body axes. Thus

$$\begin{aligned} u_\infty &= V_\infty \cos(\beta)\cos(\alpha) \\ v_\infty &= -V_\infty \sin(\beta) \\ w_\infty &= V_\infty \cos(\beta)\sin(\alpha) \end{aligned} \tag{3.2}$$

The $C_L$, $C_D$, and $C_S$ force coefficients in file **fomoco.out** are relative to the wind axes, rather than body axes. Thus body forces $F_x$, $F_y$, $F_z$ are resolved into lift, drag and side force as

$$\begin{aligned} L &= -F_x \sin(\alpha) + F_z \cos(\alpha) \\ D &= \left[ F_x \cos(\alpha) + F_z \sin(\alpha) \right] \cos(\beta) - F_y \sin(\beta) \\ S &= \left[ F_x \cos(\alpha) + F_z \sin(\alpha) \right] \sin(\beta) + F_y \cos(\beta) \end{aligned} \tag{3.3}$$

Roll, pitch, and yawing moments are listed as positive or right-hand rotation in the inertial coordinate system. No effort is made to change signs to correspond to stability axes (i.e., $x$ pointing forward, $z$ down). Moment coefficients are computed about the moment reference center specified in the **mixsur.fmp** input file. This location should be consistent with the position of the grids that make up the body in the **grid.in** input file. For moving-body simulations (OVERFLOW-D-mode) the moment reference center will move with the body. Reference areas and lengths for the force and moment coefficients are also taken from the **mixsur.fmp** file**.** The reference Mach number (**REFMACH**) and Reynolds number (**REY**) are taken from the **&FLOINP** NAMELIST input. The reference Mach number (**REFMACH**) is set to the free-stream Mach number (**FSMACH**) if the reference Mach number is not specified.

## 3.3 NAMELIST Inputs

This section outlines the format for the NAMELIST input file used for OVERFLOW. The default name for the NAMELIST file is **over.namelist**. Default values for each of the inputs are given in brackets at the end of the description. If the value for any NAMELIST variable is left out of the input file, it is automatically set to the default value. A number of the NAMELISTs are repeated for each grid. If an input variable is set for one grid, generally it becomes the default for all following grids, until the variable is set again. This is true for all variables (such as numerical scheme, smoothing parameters, etc.) that are not dependent on specific grid topology. Notable exceptions are boundary conditions, turbulent regions, or enabling viscous terms in specific coordinate directions. The user must define the beginning and end of each NAMELIST (e.g. **&NAMELIST ... /**) even if only the default values are required. Fig. 3.1 shows the names of the NAMELISTS for OVERFLOW 2.2. A full description of the NAMELIST input variables is given in Appendix B.

*Terminology:* in this manual, the Fortran standard NAMELIST terminology of "**&NAMELIST ... /**" is used, even though many systems still accept the traditional form of "**$NAMELIST ... $END**". In fact, many of the OVERFLOW test cases still use this form, and occasional lapses may be found in this document.

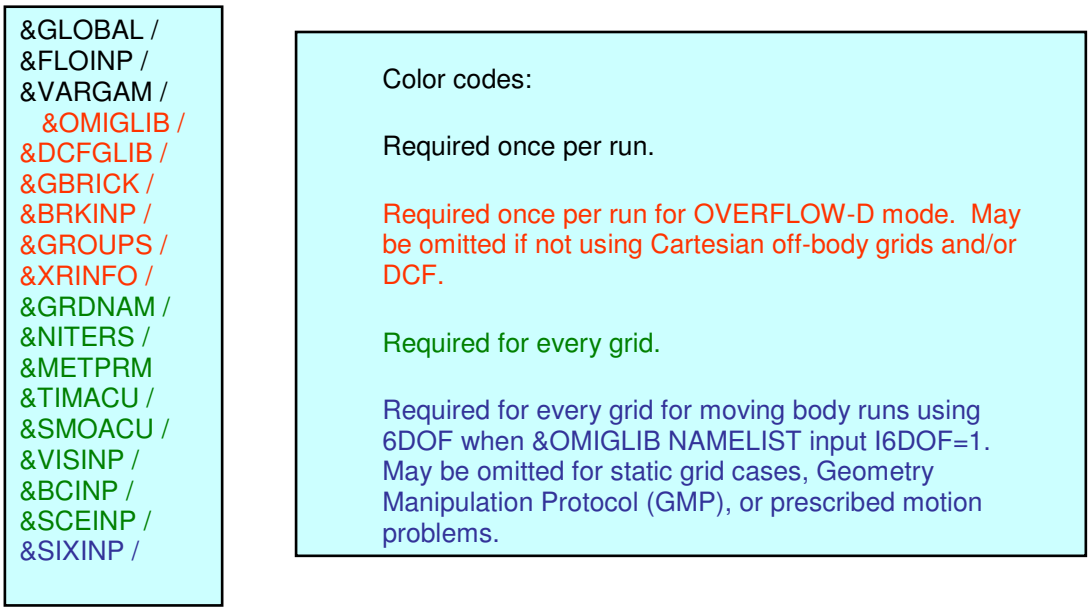



Figure 3.1 NAMELIST inputs for OVERFLOW 2.2.

Most OVERFLOW applications can be run using default values of the input variables. As an example, the input required to run a 2D inviscid airfoil on an "O" topology grid using the code defaults would be:

**Example Input 3.1**

```
&GLOBAL  NSTEPS = 500,
/
&FLOINP  ALPHA  = 2.0, FSMACH = 0.8,
/
&GRDNAM  NAME   = 'WING',  /
&NITERS /
&METPRM /
&TIMACU /
&SMOACU /
&VISINP /
&BCINP
      IBTYP  =  1, 47, 10, 21,
      IBDIR  =  2, -2,  1,  3,
      JBCS   =  1,  1,  1,  1,
      JBCE   = -1, -1,  1, -1,
      KBCS   =  1, -1,  1,  1,
      KBCE   =  1, -1, -1, -1,
      LBCS   =  1,  1,  1,  1,
      LBCE   = -1, -1, -1,  1,
/
&SCEINP /
```

Several examples of NAMELIST input files can be found in the test cases that are included in the OVERFLOW 2.2 distribution. There are some combinations of input that are frequently used. The following are some examples of the more common input combinations.

**Notes on Newton or Dual-Time Sub-Iterations**

Newton or dual-time sub-iterations are used to improve the accuracy of unsteady simulations, and can also improve the robustness of steady or unsteady simulations. The sub-iterations improve the solution accuracy near interpolated and extrapolated boundaries and also reduce the global solution error at a given time step. The solution is advanced one physical time step (**DTPHYS**) at the end of each sub-iteration cycle when Newton or dual-time sub-iterations are used. In the following example 5 dual-time sub-iterations are used with second-order time advancement. Note that the computational time step (**DT**) does not have to equal the physical time step (**DTPHYS**). Also, the computational time step (**DT**) is non-dimensionalized by the free-stream speed of sound, while the physical time step (**DTPHYS**) is non-dimensionalized by the reference velocity.

**Example Input 3.2**

```
&GLOBAL
      NSTEPS = 500,
      DTPHYS = 1.0, NITNWT = 5,
/
...
&TIMACU
      ITIME  = 1, DT = 0.1, CFLMIN = 10,
/
```

Newton sub-iterations are similarly specified using the following input. Note that the time step **DT** is not set (or is set to 0), which causes it to be set to match the physical time step (**DTPHYS**).

**Example Input 3.3**

```
&GLOBAL
      NSTEPS = 500,
      DTPHYS = 1.0, NITNWT = 5,
/
...
&TIMACU
      ITIME  = 0, DT = 0.0, /
```

**Notes on Grid Sequencing**

Grid sequencing can be used to improve convergence by initially running the solution on coarser grids. This allows the solution to set up quickly. Note that no restart files are written during the grid sequencing portion of the calculation, but forces on a body will be calculated during the grid sequencing process. The NAMELIST input for grid sequencing for 150 iterations on a coarse-grid level, 150 iterations on a medium-grid level, and 500 iterations on the fine-grid level would be

**Example Input 3.4**

```
&GLOBAL
      NSTEPS = 500,
      FMG = .TRUE., FMGCYC = 150,150, NGLVL = 3,
/
```

If the NAMELIST input **FMG** is set to TRUE the code will perform grid sequencing – even on a restart of the code. Hence **FMG** should be set to false if grid sequencing is not desired.

**Notes on Multigrid**

Multigrid is another convergence acceleration technique. In a multigrid algorithm the solution update vector ($\Delta q$) is updated with contributions from coarser grid levels at each time step. This allows low frequency error waves to be convected rapidly out of the computational domain. Multigrid is generally used for steady state problems. The input for a 3-level multigrid solution would be

**Example Input 3.5**

```
&GLOBAL
      NSTEPS = 500,
      MULTIG = .TRUE., NGLVL = 3,
/
```

When combined with grid sequencing, the multigrid algorithm uses at most **NGLVL** grid levels. Thus multigrid is not used on the coarsest-grid level, and is used on reduced levels on medium-grid levels. For example, the following input will run 150 iterations on the coarse level, 150 iterations (with 2-level multigrid) on the medium level, and 500 iterations (with 3-level multigrid) on the fine-grid level.

**Example Input 3.6**

```
&GLOBAL
      NSTEPS = 500,
      MULTIG = .TRUE., FMG = .TRUE., FMGCYC = 150,150, NGLVL = 3,
/
```

**Notes on Implicit Solvers**

The implicit solvers **ILHS**=5 and 6 use Steger-Warming flux-split (upwind) Jacobians, and can be used with any of the right hand side options in the code. However, if low-Mach number preconditioning is selected, they will revert to the central difference flux Jacobians. The default values for the central difference implicit solvers **ILHS**=0 and 2 are set assuming they are to be used with the central difference right-hand side (**IRHS**=0). These implicit solvers can be used in conjunction with upwind right-hand side algorithms (**IRHS**=2-5) if the NAMELIST parameters are changed from the default values. The recommended smoothing values for using central difference implicit schemes with upwind right hand sides are

**Example Input 3.7**

```
&METPRM
      IRHS = 5, ILHS = 2, IDISS = 2,
/
...
&SMOACU
      DIS2 = 10.0, DIS4 = 0.1, SMOO = 0.0,
/
```

**Notes on Low-Mach Number Preconditioning**

Details of the low-Mach number preconditioning scheme used in OVERFLOW 2.2 are given in Chapter 1 and in Ref. 5. Not all of the solution algorithms in the code can currently be used with low-Mach number preconditioning. The central difference (**IRHS**=0), Roe upwind (**IRHS**=4), HLLC and WENO upwind (**IRHS**=5) schemes are all compatible with preconditioning. The two central difference implicit solvers (**ILHS**=0,2) and the D3ADI solver (**ILHS**=4) are compatible with preconditioning, as are the implicit solvers **ILHS**=5,6 (by reverting to central differencing). The NAMELIST input parameter **BIMIN** ($\beta_{min}$ in Eq. (1.15), Chapter 1) in **&METPRM** is used to control the low-Mach number preconditioning. If **BIMIN**=1 the preconditioning is disabled; if **BIMIN**=-1 the preconditioning is enabled and the preconditioning factor is set to $3M_{ref}^2$. If 0 < **BIMIN** < 1.0 the preconditioning is enabled and the input value is used for the preconditioning constant. An example of input for a low-Mach number preconditioned run is:

**Example Input 3.8**

```
&METPRM
      IRHS = 5,  ILHS = 2,  IDISS = 2,  BIMIN = -1,
/
...
&SMOACU
      DIS2 = 10.0, DIS4 = 0.1, SMOO = 0.0,
/
```

For unsteady flows with preconditioning, dual time stepping must be used. However, effective use of low-Mach preconditioning with dual time stepping requires careful setting of BIMIN, as discussed in Ref. 6.

**Notes on 5th-Order WENO Schemes**

Two upwind 5th-order interpolation schemes can be run in conjunction with the AUSM flux scheme (**IRHS**=3), the Roe flux scheme (**IRHS**=4), or the HLLC flux scheme (**IRHS**=5). All of these schemes require that the spatial order be set to 5th-order (**FSO**=5). The 5th-order spatial WENO scheme is specified using the following NAMELIST inputs:

**Example Input 3.9**

```
&METPRM  IRHS = 5,
/
...
&SMOACU  FSO = 5,
/
```

Note that **ILIMIT** input parameter is not specified in Example Input 3.9. The WENO scheme will be run if **IRHS**=3, 4, or 5 and **FSO**=5, and **ILIMIT** is set to anything but 4. The WENOM scheme will be run if **IRHS**=3, 4, or 5, **FSO**=5, and **ILIMIT=**4 as shown in the following example.

**Example Input 3.10**

```
&METPRM  IRHS = 5, ILIMIT = 4,
/
...
&SMOACU  FSO = 5,
/
```

## 3.4 Turbulence Model Specifications

OVERFLOW 2.2 contains several turbulence models. Each of these turbulence models also contains a number of optional corrections and capabilities. The following table contains the baseline turbulence models currently implemented in the code.

| NQT | Turbulence Model |
|---|---|
| 0 | Baldwin-Lomax[7] algebraic model |
| 100 | Baldwin-Barth[8] 1-equation model |
| 101 | Spalart-Allmaras[9] 1-equation model with trip line specification |
| 102 | Spalart-Allmaras[9] 1-equation model |

| | |
|---|---|
| 202 | k-ω[10] 2-equation model using DDADI implicit solver |
| 203 | SST[11] 2-equation model using DDADI implicit solver |
| 204 | k-ω[10] 2-equation model using SSOR implicit solver |
| 205 | SST[11] 2-equation model using SSOR implicit solver |

**Notes on Baldwin Lomax**

The Baldwin-Lomax algebraic model requires that viscous regions (described in Appendix D) be specified in the **&VISINP** NAMELIST. The start and end index specification for viscous regions can use negative numbers to indicate indices relative to the last index value. For example, -1 represents $J_{max}$ ($K_{max}$ or $L_{max}$), -2 represents $J_{max}$-1, etc. Thus a region specified as running from 1 to -1 in J will be applied from 1 to $J_{max}$. The **ITTYP** variable is set to 1 for boundary layers and to 2 for shear layers. Separate viscous regions should be specified for each wall and shear layer in the flow field. Note that the eddy viscosity will only be calculated in regions specified with the viscous regions input.

When specifying starting and ending indices for the coordinate direction normal to the applied turbulence model condition in the &**VISINP** section, the ending index indicates the end of the search region for the $F_{max}$ peak. Thus the ending index should be beyond the expected edge of the boundary layer, and will serve to cut off the search for that edge if it has not been found closer to the wall.

The various Baldwin-Lomax boundary layer models all use the same routines. A cutoff value following the work of Degani-Schiff[12] is incorporated, but left as an input variable. For this mode of operation, the search for $F_{max}$ is stopped when $F$ drops below the **CUTOFF** times the current $F_{max}$. For the standard Baldwin-Lomax model, **CUTOFF** is set to -1 (default). (Since $F$ is never negative, the search is never stopped because of the cutoff criterion.) For the standard Degani-Schiff implementation, set **CUTOFF**=0.9. Note that **CUTOFF** should be less than one. **CUTOFF** is specified as the **TLPAR1** parameter in NAMELIST **&VISINP**. The search for $F_{max}$ is stopped when a blanked-out region is encountered.

For the Baldwin-Lomax shear layer model, the parameter $C_{wk}$ may need to be modified depending on the type of free shear layer being modeled. $C_{wk}$ is specified as the **TLPAR1** parameter in NAMELIST **&VISINP**. The default value is 2. The search for the center of the shear layer skips over blanked-out regions.

When more than one Baldwin-Lomax turbulence region is specified and the viscous regions are allowed to overlap, the maximum of the eddy viscosities from all the regions at each point is used in the viscous flux calculations. An example NAMELIST for a channel flow with a k dimension of 101 is

**Example Input 3.11**

```
&GLOBAL  NQT = 0,
/
&VISINP
      ITTYP  =  1,  1,
      ITDIR  =  2, -2,
      JTLS   =  1,  1,
      JTLE   = -1, -1,
      KTLS   =  1, 51,
      KTLE   = 51, -1,
      LTLS   =  1,  1,
      LTLE   = -1, -1,
/
```

**Notes on Setting Boundary Layer Transition Location**

The viscous regions input (Appendix D) can also be used to turn off the production terms for the Baldwin-Barth, Spalart-Allmaras, k-ω, and SST models in the specified region of the flow. This can be used to force boundary layer transition at a given point in the flow. This methodology does not accurately simulate the actual boundary layer transition process – but it does provide a convenient method for assessing the effect of transition location. This is done by setting **ITTYP=**102 in the **&VISINP** NAMELIST. The following is an example for forcing transition at the j=15 point in the grid.

**Example Input 3.12**

```
&VISINP
      ITTYP  =102,
      ITDIR  =  2,
      JTLS   =  1,
      JTLE   = 15,
```

```
      KTLS   =  1,
      KTLE   = -1,
      LTLS   =  1,
      LTLE   = -1,
/
```

**Notes on Specifying Trip Line Location**

The viscous regions (Appendix D) are also used to specify the location of trip lines for the **NQT**=101 version of the Spalart-Allmaras model. A boundary layer trip line location can be identified by specifying a viscous region with **ITTYP**=103. Trip lines should lie within regions specified as viscous wall boundary conditions. The distance from a field point to the nearest trip line determines the strength of the trip function. Distances are only computed within a grid zone, e.g., a trip line in grid 1 will not force transition in grid 2 (though transition may occur in grid 2 through convection). The following is an example for forcing transition at the j=15 line on a k=1 wall in the grid.

**Example 3.13**

```
&VISINP
      ITTYP  =103,
      ITDIR  =  2,
      JTLS   = 15,
      JTLE   = 15,
      KTLS   =  1,
      KTLE   =  1,
      LTLS   =  1,
      LTLE   = -1,
/
```

**Notes on Rotation and Curvature Corrections**

Both the Spalart-Allmaras and SST models also have options for applying corrections for rotation and curvature[13]. These corrections can be used to improve the solution accuracy for flows with strong curvature effects or for vortical flows. Two corrections are included in OVERFLOW 2.2. These corrections are invoked by specifying the **IRC** input in NAMELIST **&VISINP**.

**Notes on Compressibility and Temperature Corrections to the SST Model**

The SST model also includes corrections for compressibility[14] and temperature[15]. Compressibility tends to reduce the mixing in a flow. Compressibility effects on the turbulence model begin at about a local Mach number of one. The compressibility correction is controlled with the **ICC** input in NAMELIST **&VISINP**. The compressibility correction should normally be included with this model and this is the default. The temperature correction is controlled with the **ITC** input in NAMELIST **&VISINP**. The temperature correction can be used when simulating the mixing of two streams with greatly differing total temperature values. A total temperature gradient will cause an increase in the mixing of the two streams. This correction should be used with care, and is off by default.

**Notes on DES and DDES Hybrid RANS/LES Models**

The Spalart-Allmaras and SST models also include options for running these models as hybrid RANS/LES models. The detached eddy simulation (DES[16]) and delayed detached eddy simulation models (DDES[16]) models are available for the Spalart-Allmaras and SST models. The SST Multi-Scale model[17] is also available. These models are invoked by specifying the **IDES** input in NAMELIST **&VISINP**. These models can also be used with the rotation and curvature corrections, compressibility corrections, and temperature corrections.

## 3.5 Variable γ and Multispecies

OVERFLOW 2.2 supports variable γ and multispecies calculations. The **IGAM** input parameter in NAMELIST **&VARGAM** is used to specify the type of gas model desired. The default gas model is a constant γ specified by setting **IGAM**=0. The value for γ can be set by the **GAMINF** input parameter in NAMELIST **&FLOPRM**. The default value for γ is 1.4.

*Important note:* the use of variable gas properties (i.e., γ as a function of temperature) is not implemented in a fully correct manner, and is not recommended at this time. This includes any setting of **ALT1-ALT4** or **AUT1-AUT4**≠0. The simulation of flows with multiple species is handled correctly as long as each species uses a constant γ.

**Notes on Single Gas with Varying Gas Properties**

If **IGAM**=1 the code assumes a single gas with the variation of $c_p/R$ (or $\gamma/(\gamma\text{-}1)$) given by

$$\frac{c_p}{R} = \frac{\gamma}{\gamma - 1} = a_0 + a_1 T + a_2 T^2 + a_3 T^3 + a_4 T^4 \tag{3.4}$$

The following is an example input for a variable $\gamma$ single species simulation.

**Example Input 3.14**

```
&VARGAM
      IGAM = 1,
      ALT0 = 3.653,
      ALT1 =-0.1334E-2,
      ALT2 = 0.3293E-5,
      ALT3 =-0.1913E-8,
      ALT4 = 0.2763E-12,
      AUT0 = 3.045,
      AUT1 = 0.1338E-2,
      AUT2 =-0.4883E-6,
      AUT3 = 0.8556E-10,
      AUT4 =-0.5702E-14,
/
...
&SCEINP /
```

No species equation is required for this option, so no input is specified for the **&SCEINP** NAMELIST. The **ALT0-ALT4** inputs are for the temperature range ($540^oR \le T \le 1800^oR$), and **AUT0-AUT4** inputs are for the range ($1800^oR \le T \le 9000^oR$).

**Notes on 2-Species Variable $\gamma$**

A special option exists for simulating flows with two species, where the values of stagnation enthalpy of the two streams are significantly different. In this case, the stagnation enthalpy is used to determine the species concentration, rather than solving separate species equations. This option is particularly useful for rocket plumes, and is selected by setting **IGAM**=2. Sample input for a 2-species simulation using the **IGAM**=2 option is given below.

**Example Input 3.15**

```
&VARGAM
      IGAM  = 2,
      HT1 = 2.0, HT2 = 5.0,
      SCINF =  1.0,   0.0,
      SMW   = 28.97, 20.13,
      ALT0  =  3.50,  4.85,
/
...
&SCEINP /
```

In this case we let **ALT1-ALT4** and **AUT1-AUT4** default to 0 so as to simulate two gases each with constant $\gamma$. The free-stream species concentrations for gas 1 and gas 2 are given by **SCINF**; here the free-stream is comprised of all gas 1. The molecular weights for the two gases are specified by **SMW**, and can be given as absolute values or as ratios to the free-stream molecular weight. The molecular weights are used to compute the gas constant $R$ for each species. The stagnation enthalpy ratios **HT1** and **HT2** are used to determine the species concentrations. **HT1** is the value of $h_0/h_{0_\infty}$ below which the mixture is assumed to be all gas 1, while **HT2** is the value above which the mixture is assumed to be all gas 2. In between, the species concentration is a linear function of the local stagnation enthalpy ratio.

**Notes on Multi-Species**

Finally, the following input is an example where the species equation solver is used to simulate a flow with two species. Note that the species equations are solved loosely coupled with each other and with the mean flow

equations. The use of Newton or dual-time step sub-iterations improves the coupling since information is shared between each sub-iteration.

**Example Input 3.16**

```
&GLOBAL  NQC = 2,  /
&VARGAM
      SCINF =  1.0,   0.0,
      SMW   = 28.97, 20.13,
      ALT0  =  3.50,  4.85,
      SIGL  =  1.0,   1.0,
      SIGT  =  1.0,   1.0,
/
&SCEINP
      ITLHIC = 10, IUPC = 2, FSOC = 3,
/
```

**SIGL** and **SIGT** are the laminar and turbulent Schmidt numbers, respectively, for each species.

OVERFLOW 2.2 has three options for the species convection terms: central differencing (**IUPC**=0), upwind differencing (**IUPC**=1), and the HLLC upwind differencing (**IUPC**=2). Central differencing is always $2^{nd}$-order in space, while upwind differencing may be $1^{st}$-, $2^{nd}$-, or $3^{rd}$-order in space, and HLLC may be $1^{st}$-, $2^{nd}$-, or $3^{rd}$- or $5^{th}$-order in space. The species equations may be implicitly solved using an ADI method (**ITLHIC**=1) or using an SSOR solver (**ITLHIC**>1). **ITLHIC** specifies the number of iterations to use in the matrix solution process. Ten iterations are recommended for the SSOR method. The utility code **vgplot** (see Chapter 7) can be used to post-process the OVERFLOW 2.2 output.

OVERFLOW recalculates the value of free stream $\gamma$ using the reference temperature and species in NAMELIST **&FLOPRM** when **IGAM**=1 or 2 or if multiple species are present. The new value of **GAMINF** is used in the code and written to the **q.save** restart file. The input value of **GAMINF** is ignored.

## 3.6 Boundary Condition Specification

All boundaries must have a boundary condition specified in the **&BCINP** NAMELIST, with the exception of interpolated boundaries. As with viscous regions, the start and end index specification for boundary conditions can use negative numbers to indicate indices relative to the last index value. For example, -1 represents $J_{max}$ ($K_{max}$ or $L_{max}$), -2 represents $J_{max}$-1, etc. Thus a region specified as running from 1 to -1 in J will be applied from 1 to $J_{max}$. When specifying a boundary condition region, it is expected that the start and end indices will be equal in the coordinate direction normal to the applied boundary condition (**IBDIR**). The only exceptions to this rule are prescribed Q (**IBTYP**=42), blanked-out region (**IBTYP**=61), copy-to/copy-from (**IBTYP**=70/71), vortex generator vane model (**IBTYP**=601), and the unsteady flow output region (**IBTYP**=201). **Boundary conditions are applied in the order in which they are specified.** A list of boundary conditions supported in OVERFLOW 2.2 is given in Appendix C.

**Notes on Wall BC's IBTYP = 1-9**

Wall boundary conditions (**IBTYP**=1-9) use a slow start if the free stream Mach number is above 0.5. The slow start ramps the imposition of the boundary condition from time step 0 to time step 30. For the constant temperature wall boundary conditions (**IBTYP**=3,4,7,8), $T_{wall}$ (degrees Rankine) is specified by setting **BCPAR1** in NAMELIST **&BCINP**. **IBTYP**=9 gives a viscous adiabatic wall with rotation. **BCPAR1** specifies the rotation rate (radians per unit time, non-dimensionalized by $V_{ref}$), and **BCPAR2**=1,2,3 gives the axis of rotation (x, y, or z).

**Notes on Periodic BC's IBTYP = 10,18**

Two types of periodic boundaries are supported in OVERFLOW 2.2. **IBTYP**=10 is used for grids such as bodies of revolution where the first and last planes for a given index are identical in space. The boundary condition does not require any additional grid planes. **IBTYP**=18 may be used for periodic boundaries such as an infinite channel where the first and last plane are identical, but not co-located in space. Both boundary conditions need only be applied to one plane (normally the J, K, or L=1 plane). The boundary condition will update both planes in the solution process. All periodic boundaries are treated implicitly inside the code.

**Notes on Symmetry BC's IBTYP = 11-13,17**

The symmetry boundary conditions **IBTYP**=11,12,13 require a reflection plane be included in the grid. The flow equations will be solved on the symmetry plane and the boundary condition is applied on the reflection plane.

The symmetry boundary condition **IBTYP**=17 does not require a reflection plane be included in the grid. This boundary condition is a slip wall with second-order extrapolation, and hence the flow equations are not solved on the symmetry plane. The slip wall boundary conditions (**IBTYP**=1 and 2) can also be used for a symmetry boundary condition.

**Notes on Polar Axis BC's IBTYP = 14-16**

The polar axis boundary conditions (**IBTYP**=14,15,16) extrapolate values to the axis to set the derivative to zero at the axis. The value at the axis is computed as

$$f_0 = f_1 + \frac{\alpha}{3}(f_1 - f_2) \tag{3.5}$$

where $f_0$ is the desired value at the axis, and $f_n$ the value at the $n$th point from the axis. Setting $\alpha$=0 results in $0^{th}$-order extrapolation, while $\alpha$=1 reflects a $1^{st}$-order condition. The value of $\alpha$ is specified by setting **BCPAR1** in NAMELIST **&BCINP**. The default value for $\alpha$ is 1. The most stable value for $\alpha$ is 0. Once values are extrapolated for all points around the axis, they are averaged to obtain the final value used on the axis.

**Notes on 2D BC IBTYP = 21**

The 2D boundary condition **IBTYP**=21 assumes that the flow is in the *x*-*z* plane. The grid should contain three constant *y*-planes with *y*=-1, 0, and 1. The best code efficiency occurs when the L-index is used for the *y*-planes ($L_{max}$=3). This boundary condition may be applied to the first or last index plane.

**Notes on the Axisymmetric BC IBTYP = 22**

The axisymmetric boundary condition **IBTYP**=22 assumes that the flow is in the *x*-*z* plane. The grid should contain three planes with the first and last planes rotated +/-$1^o$ from the center plane. The best code efficiency occurs when the L-index is used for the *y*-planes ($L_{max}$=3). This boundary condition may be applied to the first or last index plane.

**Notes on Supersonic/Subsonic Inflow/Outflow BC IBTYP = 32**

The supersonic/subsonic inflow/outflow boundary condition (**IBTYP**=32) does the following: for supersonic inflow, boundary values are left at their current state; for subsonic inflow, a characteristic condition based on Riemann invariants is applied (also referencing existing flow conditions on the boundary); for subsonic outflow, pressure is held constant while other flow quantities are extrapolated; and for supersonic outflow, all variables are extrapolated. Note that this condition typically uses some or all values already on the boundary. Other boundary conditions, however, use this condition in combination with fixing certain boundary values.

*General advice:* use **ITBYP**=47 as a "general-purpose" far-field boundary condition instead of 32. BC 32 is subject to corruption of flow conditions on the boundary due to transient flow during convergence (for example, the boundary becoming outflow temporarily). BC 32 can also exhibit glitches where a far-field boundary switches from inflow to outflow, due to the different types of boundary conditions imposed.

**Notes on Freestream/Characteristic Inflow/Outflow BC IBTYP = 47**

This boundary condition imposes free-stream conditions on the boundary, then applies a characteristic condition, taking information from the boundary or the interior of the domain, as appropriate, based on the velocity component normal to the grid boundary. This boundary condition is appropriate for use as a general-purpose far-field condition.

**Notes on Specified Pressure Outflow BC IBTYP = 33**

The specified pressure outflow condition (**IBTYP**=33) fixes pressure and extrapolates other flow quantities. This boundary condition can be used to control mass flow for internal flows. Mass flow can be adjusted by changing the specified pressure value. The outflow pressure $p/p_\infty$ is specified by setting **BCPAR1** in NAMELIST **&BCINP**.

**Notes on Specified Mass Flow BC's IBTYP = 34,36**

The mass flow boundary conditions **IBTYP**=34,36 attempt to drive to a specified mass flow rate by adjusting the exit pressure for an internal flow. For **IBTYP**=34, the pressure will be a constant across the exit face, while for **IBTYP**=36, the pressure is allowed to vary. The current flow rate predicted by the code is calculated using the

FOMOCO subroutines, allowing the integration plane to be composed of multiple grids. The target mass flow is specified by setting **BCPAR1** in NAMELIST **&BCINP**. **BCPAR2** is used to specify the number of time steps between updates of the exit pressure and the relaxation value for the exit pressure. This is done to improve the convergence and robustness of this boundary condition. The relaxation value varies between 0 and 1. **BCPAR2**=10.5 would tell the code to update the pressure every 10 time steps with a relaxation value of 0.5. The default value for **BCPAR2** is 1.2 (update every iteration with 0.2 relaxation factor). **BCFILE** is used to specify the name of the component surface in the FOMOCO input file to be integrated to get the mass flow. The reference area ($A_{ref}$) will also be obtained from the specified FOMOCO input file for the component surface name specified in **BCFILE**. The target non-dimensional mass flow ( $\overline{\dot{m}}$ ) is defined as

$$\overline{\dot{m}} = \frac{\dot{m}}{\rho_\infty U_\infty A_{ref}} \tag{3.6}$$

The mass flow used with turbine engines is often specified in terms of the "corrected mass flow" ( $\dot{m}_c$ ), where a correction is used to scale the mass flow to standard day sea-level conditions. The corrected mass flow is defined as

$$\dot{m}_c = \dot{m}\frac{\sqrt{\theta}}{\delta} \tag{3.7}$$

where

$$\theta = \frac{T_{t\infty}}{T_{sd}},\ \delta = \frac{P_{t\infty}}{P_{sd}} \tag{3.8}$$

Here the standard day temperature ($T_{sd}$) is taken as 518.67°R and the standard day pressure ($p_{sd}$) is 14.6958 psi. The target non-dimensional mass flow for this boundary condition using corrected mass flow can then be written as

$$\overline{\dot{m}} = \frac{\dot{m}_c\left(1+\frac{(\gamma_\infty-1)}{2}M_\infty^2\right)^{\frac{(\gamma_\infty+1)}{2(\gamma_\infty-1)}}}{\frac{p_{sd}}{\sqrt{T_{sd}}}M_\infty A_{ref}\sqrt{\frac{\gamma_\infty}{R_\infty}}} = \frac{\dot{m}_c\left(1+\frac{(\gamma_\infty-1)}{2}M_\infty^2\right)^{\frac{(\gamma_\infty+1)}{2(\gamma_\infty-1)}}}{\rho_{sd}a_{sd}M_\infty A_{ref}\sqrt{\frac{\gamma_\infty R_{sd}}{\gamma_{sd}R_\infty}}} \tag{3.9}$$

where $\rho_{sd}$ is standard day density (0.07657 lbm/ft$^3$) and $a_{sd}$ is the standard day speed-of-sound (1117 ft/sec). $R_{sd}$ and $R_\infty$ are the standard and reference gas constants respectively.

**Notes on Inflow BC IBTYP = 41, 141, 143**

The inflow boundary conditions **IBTYP**=41, 141, and 143 allow total pressure ($p_t/p_{t\infty}$) and total temperature ($T_t/T_{t\infty}$) to be specified using **BCPAR1** and **BCPAR2**, respectively. All of these boundary conditions use a Riemann extrapolation of static conditions with the total conditions to determine the inflow values. **IBTYP**=41 does this on a point-by-point basis. Existing velocity direction on the boundary face is used, and velocity magnitude may vary across the face. **IBTYP**=141 averages over the face and sets uniform conditions on the boundary with a velocity normal to the face. These boundary conditions are often used for nozzle inflow cases since they can allow the mass flow to vary during the convergence process. Note that the entire face must be included in a single grid for the averaging in boundary condition **IBTYP**=141 to work properly. **IBTYP**=143 is the same as 141, except that **BCFILE** is used to specify a FOMOCO component for averaging of static flow conditions and surface normal.

**Notes on Prescribed Q BC IBTYP = 42 & 45**

Boundary conditions **IBTYP**=42 and 45 read the Q flow variables from a file. The Fortran unformatted boundary condition file has the format of a (single-grid) OVERFLOW Q restart file:

```
READ(10) J1,K1,L1
READ(10)
READ(10) Q(JS:JE,KS:KE,LS:LE,1:NQ)
```

Here the Q header information, if present, is ignored. Dimensions in the Q file (J1,K1,L1) must match the computational region over which the boundary condition is applied, i.e., J1=JE-JS+1, etc. The file is read every iteration.

A starting step (iteration) number for the boundary condition can be specified in **BCPAR1**. This allows a converged solution to be obtained, then a plume exit condition to be imposed. Further, a slow-start may be imposed by setting **BCPAR2** to the number of steps to ramp up the boundary condition. Boundary condition **IBTYP**=45 reads Q from a file, then imposes the inflow/outflow condition used in **IBTYP**=32.

**Notes on Actuator Disk BC IBTYP = 44**

The actuator disk (**IBTYP**=44) is modeled by averaging Q variables across the disk, then adjusting pressure in planes adjacent to the disk to impose the required $\Delta p$. The pressure jump $\Delta p/p_\infty$ is specified by setting **BCPAR1** in NAMELIST **&BCINP**. **IBDIR** indicates the flow direction.

**Notes on Jet Mass Flow BC IBTYP = 48 & 148**

Boundary condition **IBTYP**=48 and 148 are simple jet mass flow boundary conditions for steady and unsteady jets. The steady condition (**IBTYP**=48) uses **BCPAR1** to specify the mass flow ratio $(\rho V)_{jet}/(\rho V)_\infty$. The unsteady condition (**IBTYP**=148) uses **BCPAR1** to specify the flow solver step number which corresponds to time $t$=0. The file **BCFILE** includes four numbers, the minimum ($X_{min}$) and maximum ($X_{max}$) mass flow rates, and the frequency $f$ and phase angle $\phi$ of the sinusoidally varying jet strength. The time-varying mass flow is scaled by

$$M_{scale} = \frac{X_{\max} + X_{\min}}{2} + \frac{X_{\max} - X_{\min}}{2}\cos\left(2\pi f t + \phi\right) \tag{3.10}$$

The format for the formatted file for **IBTYP**=148 is

```
READ(20,*) XMIN,XMAX,XFREQ,XPHASE
```

For both conditions, stagnation enthalpy and pressure are extrapolated from the interior of the domain.

**Notes on the Blanked Region BC IBTYP=61**

A region of a grid may be blanked out using the blanked region boundary condition (**IBTYP**=61). If this boundary condition is used in conjunction with the DCF overset grid assembly the code will attempt to interpolate the fringe points surrounding the blanked region to the level specified by the NAMELIST **&OMIGLB** input **LFRINGE**. The default is double fringe.

**Notes on Slotted Wind Tunnel Wall BC IBTYP = 82**

For the slotted wind tunnel wall boundary condition (**IBTYP**=82), the factor $R$ is specified by setting **BCPAR1** in NAMELIST **&BCINP**. Typically, $R$ is set to 19 times the wall porosity ratio. For example, for the NASA Ames 11' x 11' wind tunnel with a 6% wall porosity, $R$=19x0.06 is used. See Ref. 18. *At present, the tunnel walls are expected to be parallel to the free-stream flow.*

**Notes on Wind Tunnel Exit BC IBTYP = 86**

For the wind tunnel exit condition (**IBTYP**=86), **BCPAR1** sets the area ratio $A_{exit}/A_{ref}$. Density and pressure are extrapolated to the exit plane, and the velocity is set (to a constant across the face), such that the mass flow is equal to a free stream flow through the reference area. Velocity direction is assumed to be the same as the free stream.

Note that while this boundary condition ensures that the desired mass flow is pulled through the boundary condition face, it destroys any flow features (such as boundary layers) that are present. It should thus be applied far from regions of interest in the flow.

## 3.7 Unsteady Flow Output Options

Two options for extracting information from unsteady flow calculations for post-processing have been incorporated into OVERFLOW 2.2. The first option is to output the current values of the solution variables for a specified region of the flow. This is accomplished through specifying a region using boundary condition **IBTYP=**201. The output for any grid may be a point, line, surface, or volume subset of the grid. The output is written to a Fortran unformatted output file. The file format can be found in Appendix A. This file is not deleted during restarts, and output from subsequent runs is appended to the file. Unsteady output files are not written for coarse mesh solutions during grid sequencing. **BCPAR1** can be used to specify the iteration number to start saving flow information, and **BCPAR2** can specify the iteration increment for saving. Information will be saved when *mod*(step#,BCPAR2)=0. The output file will be written to the file name specified by **BCFILE**. If no name is specified, the file will be written to the file **BC_201.n.ib** where *n* is the grid number and *ib* is the BC number.

For the second option, it is sometimes desirable to obtain a time-averaged flow field for the entire grid system during an unsteady flow simulation. This can be accomplished by setting the **ISTART_QAVG** input variable in the **&GLOBAL** NAMELIST to the iteration number when the user would like to begin averaging. A file called **q.avg** will be output using a modified PLOT3D Q file format, described in Appendix A.

The following example input is for a generic weapons bay. This input file uses both unsteady output options. The grid system has two grids, a flat plate and the bay. The time-averaging of the unsteady flow field begins after iteration 2300. The Q variables for each time step are extracted on the bay centerline using BC 201, for later Fourier analysis.

This example also uses the DCF grid assembly, which will be described in Chapter 4. Triple-fringe interpolations are used in conjunction with $4^{th}$- and higher-order inviscid flux schemes.

**Example Input 3.17**

```
&GLOBAL
      NSTEPS = 12300, RESTRT = .F.,
      MULTIG = .F., FMG = .T., FMGCYC = 150,150,
      DTPHYS = 0.1, NITNWT = 5,
      NQT    = 102,
      ISTART_QAVG = 2300,
/

&OMIGLB  LFRINGE = 3,
/
&DCFGLB  DQUAL = 0.05,
/
&GBRICK  OBGRIDS = .F.,
/
&BRKINP /
&GROUPS /

&FLOINP
      FSMACH = 0.95, REY = 2.083E5, TINF = 465.9,
/
&VARGAM /

&GRDNAM  NAME = 'plate',  /
&NITERS /
&METPRM
       IRHS = 5, ILIMIT = 3, ILHS = 6,
/
&TIMACU  ITIME = 0,
/
&SMOACU  FSO = 5,
/
&VISINP
      WALLFUN = .T., FSOT = 1,
      IDES = 2, IRC = 0,
/
&BCINP
      IBTYP  =   1,  5,  5,  5,  5, 40, 30, 47, 47, 47,
      IBDIR  =   3,  3,  3,  3,  3,  1, -1,  2, -2, -3,
      JBCS   =   1, 11, 41, 41,161,  1, -1,  1,  1,  1,
```

```
      JBCE   =  10, 41,161,161, -1,  1, -1, -1, -1, -1,
      KBCS   =   1,  1,  1, 81,  1,  1,  1,  1, -1,  1,
      KBCE   =  -1, -1, 31, -1, -1, -1, -1,  1, -1, -1,
      LBCS   =   1,  1,  1,  1,  1,  1,  1,  1,  1, -1,
      LBCE   =   1,  1,  1,  1,  1, -1, -1, -1, -1, -1,
/

&SCEINP /
&SIXINP /

&GRDNAM  NAME = 'bay',  /
&NITERS /
&METPRM /
&TIMACU /
&SMOACU /
&VISINP /
&BCINP
      IBTYP  =   5,  5,  5,  5,  5, 201,201,201,
      IBDIR  =   3,  1, -1,  2, -2,   1, -1,  3,
      JBCS   =   1,  1, -1,  1,  1,   1, -1,  1,
      JBCE   =  -1,  1, -1, -1, -1,   1, -1, -1,
      KBCS   =   1,  1,  1,  1, -1,  26, 26, 26,
      KBCE   =  -1, -1, -1,  1, -1,  26, 26, 26,
      LBCS   =   1,  1,  1,  1,  1,   1,  1,  1,
      LBCE   =   1, 51, 51, 51, 51,  51, 51,  1,
/
&SCEINP /
&SIXINP /
```

## 3.8 Initializing the Solution

The flow will be initialized to the conditions in the **&FLOINP** NAMELIST (**FSMACH**, **ALPHA**, **BETA**, **RETINF** (for 1- and 2-equation turbulence models), and **XKINF** (for 2-equation turbulence models) when **RESTRT**=.FALSE. This is the normal mode for initializing the flow field. Boundary condition **IBTYP**=42 can also be used to initialize a boundary to some other value.

The user may also generate a restart file (**q.restart**) and use this file to initialize the flow field by setting **RESTRT**=.TRUE.. Previous solutions at different values of **FSMACH**, **ALPHA**, **BETA**, **RETINF,** or **REY** may also be used to initialize the flow field. OVERFLOW 2.2 will adjust values in the Q file for the input parameters that have been changed (i.e., rotate the velocity vector or scale the conserved variables for varying free-stream Mach number). This is often an easy way to initialize complex flow fields once an initial flow solution is obtained at one condition.

Supersonic and hypersonic flows with blunt forward-facing surfaces are often hard to initialize when using upwind flux algorithms. These upwind algorithms will generate a non-physical bow shock between the surface and the first point off the wall. This is a numerically-acceptable solution for an upwind solver in supersonic flow, since the eigenvalues of the flux system are all possitive. Two methods may be used to avoid this difficulty. The first option is to start the solution with a transonic or low supersonic free-stream Mach number. Once a region of subsonic flow is obtained in the nose region, the free-stream Mach number may be increased with successive restarts. It is often convenient to use the coarse mesh with grid sequencing during this ramp-up to expedite the process. (It is often sufficient to take one step with **FSMACH**=0.8 before resetting **FSMACH** to the desired value.) A second option is to initially run the solution using the central difference flux algorithm and switch to the upwind algorithm after subsonic flow is established in the nose region. Again it is usually good to perform these initializing runs using the coarse grid and grid sequencing. It also may be necessary to increase the $2^{nd}$-order smoothing (**DIS2**) and $4^{th}$-order smoothing (**DIS4**) for the central difference algorithm above the default values during the initializing steps. Values of **DIS2**=5, **DIS4**=0.1 are recommended.

Grid sequencing may be used to allow the proper mass flow to develop quickly on the coarse grids for internal flows as shown in the following example for a two-dimensional duct.

**Example Input 3.18**

```
&GLOBAL
      NSTEPS = 0, RESTRT = .F.,
      MULTIG = .F., FMG = .T., FMGCYC = 500,0,
```

```
      DTPHYS = 0.1, NITNWT = 5,
      NQT  = 102,
/

&FLOINP
      FSMACH = 0.3,  REY = 2.0E6, TINF = 465.9,
/
&VARGAM /

&GRDNAM  NAME  = 'duct',  /
&NITERS /
&METPRM
            IRHS = 5, ILIMIT  = 3, ILHS = 6,
/
&TIMACU  ITIME = 0,
/
&SMOACU  FSO   = 3,
/
&VISINP
      WALLFUN = .T., FSOT = 1,
/
&BCINP
      IBTYP  =   5,  5, 41, 33, 21,
      IBDIR  =   2, -2,  1, -1,  3,
      JBCS   =   1,  1,  1, -1,  1,
      JBCE   =  -1, -1,  1, -1, -1,
      KBCS   =   1, -1,  1,  1,  1,
      KBCE   =   1, -1, -1, -1, -1,
      LBCS   =   1,  1,  1,  1,  1,
      LBCE   =  -1, -1, -1, -1,  1,
      BCPAR1(3) = 1.0,
      BCPAR2(3) = 1.0,
      BCPAR1(4) = 1.0,
/
```

The solution will only be calculated on the coarse grids using this input. The inflow boundary is specified to have $p_t/p_{t\infty}$=1.0 and $T_t/T_{t\infty}$=1.0. The outflow pressure is set to $p/p_\infty$=1.0. The inflow or outflow conditions may be varied to obtain the desired mass flow for the duct. The solution can then proceed to the intermediate and fine grids.

## 3.9 Running the Code

The NAMELIST input needs to be contained in a file named **over.namelist** for both the serial and MPI versions of the code. Once the input files are generated, the serial code can be run with the following command

**./overflow**

The MPI version of the code can be run with the following command

**mpirun –np # ./overflowmpi**

The **tools/run** directory contains two scripts, **overrun** and **overrunmpi**, that can be used to execute the flow solver. The scripts allow the user to run a series of separate input files and manages the restart and output files for the user. These scripts do the following

1. Move ***.save** file to ***.restart** files before restarting
2. Highlights warnings and errors
3. Creates a log file with time/date, machine name, executable name, and NAMELIST input file name
4. Concatenates output history files upon completion

The scripts expect the NAMELIST input files to use the naming convention **basename.inp** or **basename.n.inp** where *basename* is the case name and *n* is an identifying number or text for the NAMELIST file. The script for the serial version of the code is executed with the command

**overrun basename n**

The script to run the MPI solver is run with a command that mimics the mpirun command

**overrunmpi –np <ncpus> basename n**

or

**overrunmpi –np <ncpus> -machinefile <hostfile> basename n**

Use of these run scripts is highly recommended.

The code initially performs checks on the NAMELIST input. A summary of the NAMELIST parameters is written to standard out for each mesh. If any errors in the input are encountered they are reported to standard out and the code execution halts. Next the code will perform automatic grid splitting for multi-processor runs. A summary of the grid decomposition and grouping is also written to standard out. The code will then report convergence information for each mesh to standard out every ten iterations. The code provides a summary of code timing and performance parameters at the end of the run.

The code will write a number of files during the execution. The **resid.out** text file contains the residual for each grid at each sub-iteration. The **rpmin.out** text file contains the minimum density, pressure, $\gamma$, the number of reverse flow points, the number of supersonic points, and maximum value of eddy viscosity at each iteration. The **timers.out** file is written at the completion of a run and contains timing information for OVERFLOW 2.2. This file has timing information broken down in several different ways. The **turb.out** file has the turbulence transport equation residual information when a transport turbulence model is used. The **fomoco.out** file contains forces and moments on the components identified in the **mixsur.fmp** file. The formats for these files are included in Appendix A. These files are overwritten each time the code is started; again, the **overrun** and **overrunmpi** scripts append these to the corresponding **basename.resid**, **basename.rpmin**, etc. files. Similar to the **overrun** script, the Chimera Grid Tools utility **overplot** can be executed to plot many of the OVERFLOW history files by typing

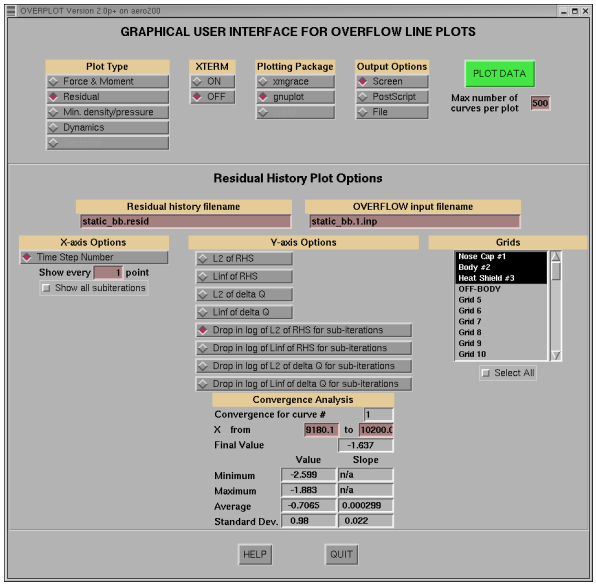
**overplot basename**

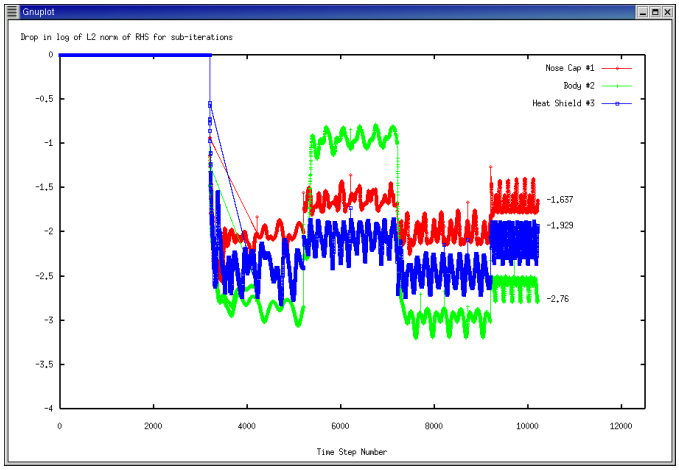
or by typing

**overplot resid.out**

This allows the user to assess convergence of the code during a run. An **overplot** example for the mean flow equation residual (**resid.out**) is shown below.

**Example 3.19 overplot utility**




The solution is written to the **q.save** restart file. The format of the restart file is in Appendix A. The restart file is written out at the time step increment given by the **&GLOBAL** NAMELIST parameter **NSAVE** (default = 100). The restart file will be written to a file named **q.#** where number is the iteration number if **NSAVE** is set to a negative number. (In this case a **q.save** file is also written on completion of the run.) The **overrun** or **overrunmpi** script automatically copies the **q.save** file to **q.restart** prior to starting the code. If OVERFLOW is run without the script, this must be done manually. Note that writing a **q.save** file can take a significant amount of time, especially for a large grid system using MPI. For efficiency, be sure that the solution is saved only as often as necessary for checkpointing or saving the run. If **NSAVE**=0, **q.save** is only written at the end of the run.

After each iteration, OVERFLOW 2.2 checks for the existence of a file named **STOP** in the current directory. If this file exists, the code will save the current flow solution and stop. This can be used to stop a run gracefully, without losing the work done so far. For example, if a job has been running for an extended length of time, one can go to the directory where the job was started from and type **touch STOP**. This creates a (zero-length) file **STOP**. When OVERFLOW 2.2 finishes the current iteration, it finds the file, saves **q.save**, and stops. As an alternative, an asci **STOP** file can be created containing an iteration number. If OVERFLOW has passed this iteration, the code will write **q.save** and stop at the end of the current iteration; otherwise it will stop at the end of the specified iteration. The **STOP** file is deleted by OVERFLOW 2.2 before exiting. The **overrun** and **overrunmpi** scripts will check for a **STOP** file before starting. If this file is found, the run will be aborted. (The contents of the file are not checked.)

OVERFLOW 2.2 also checks for a **SAVE** file following each iteration, though this check is only done during iterations on the fine-grid level. This file can be created using the **touch** command similar to the **STOP** file discussed in the previous paragraph. If the **SAVE** file is an asci file containing an iteration number, a **q.save** file is written after that iteration, and the run continues. If no number is present, or the iteration is already past, a **q.save** file is written after the current iteration.

The code may be run in parallel using MPI, OpenMP, or in a hybrid mode that uses both MPI and OpenMP. The MPI mode can be used for both distributed memory machines (PC clusters) and shared memory machines. MPI divides the work between processors based on the grid system. The grid system will be decomposed and groups of subgrids will be established for load balancing when using MPI. MPI will almost always yield good parallel performance. OpenMP is useful for shared memory machines or dual- or quad-core machines. OpenMP performs parallelization at the loop level in the code, and hence does not require grid decomposition. Not all machines or compilers will have good parallel performance when using OpenMP. The hybrid mode is useful on machines that will support both MPI and OpenMP.

The grid system is automatically decomposed when using MPI to achieve the best load-balancing possible. The number of groups for the run is set to the number of MPI processes selected for the parallel run. The default load-balancing scheme is based on equal distribution of grid points between processes (i.e., target group size). Grids are

split in half (with overlap added) until each grid is less than half the target group size. Grids are then distributed, from largest to smallest, to the current smallest group. Besides the number of grid points, each grid is additionally weighted by whether multigrid is used, and if multiple iterations are done compared to other grids (**ITER** in NAMELIST **&NITERS**). This scheme works quite well for grid systems with large numbers of grids, and reasonably well for smaller systems. Grid splitting introduces additional explicit boundaries, which affects the convergence behavior of the implicit algorithms in the code. Sub-iterations can be used to reduce the effect of the interpolated boundaries.

OVERFLOW allows the user to control the load-balancing process, but these inputs are rarely used in code execution. The **grdwghts.restart** file contains previous timing information for each grid. If the **&GLOBAL** NAMELIST parameter **GRDWTS=**.TRUE., the code will use the information in this file to perform the load balancing. If **GRDWTS=**.FALSE., the code will use the default load-balancing scheme. The input parameter **MAX_GRID_SIZE** can be used to explicitly control the splitting of grids. **MAX_GRID_SIZE=**0 selects the default grid splitting algorithm. If **MAX_GRID_SIZE>**0, the code will use the specified weighted grid size for splitting. If **MAX_GRID_SIZE<**0, grids will not be split.

In OVERFLOW-D-mode, some additional controls are available. In the **&GROUPS** NAMELIST, input parameters **MAXNB** and **MAXGRD** substitute for **MAX_GRID_SIZE**, and allow distinction between the splitting of near-body and off-body grids, respectively. The input parameter **WGHTNB** is an additional weight-factor applied to near-body grids, in case a different numerical scheme is used which significantly affects the processing of these grids. Finally, **IGSIZE** sets the maximum group size of each MPI process during grid adaptation. This allows the code to stop if the adaption procedure produces a grid system that is too large for the current number of MPI processes.

The grid splitting summary is written to standard out as shown below.

**Example 3.20**

```
Target (weighted) near-body grid size from grouping:          12862
Checking near-body grids...
Original number of near-body grids:     2
      Splitting grid     1 at K =    20
      Splitting grid     1 at K =    11
      Splitting grid     2 at K =    20
      Splitting grid     2 at K =    11
      Splitting grid     3 at J =   121
      Splitting grid     5 at J =   121
Final    number of near-body grids:     8

Target (weighted)  off-body grid size from grouping:          14752
Checking  off-body grids...
Original number of  off-body grids:    30
  Splitting grid     9 at J =    59
Final    number of  off-body grids:    31
```

Grouping information and expected parallel speedup is also reported to standard out as shown.

**Example 3.21**

```
Load balance will be based on grid size.

    Summary of work distribution for     4 groups:

    Group     Kpts      %load   Grid list
       1        30       100    4    8   11   17   14   22   21   33   31   34
                               39
       2        29        99    6    7   12   19   13   18   20   32   26   37
       3        30       100    1    9    3   23   24   28   30   35   38
       4        30       100    2   10    5   15   25   29   27   36   16

    Predicted parallel efficiency is   100%,

      based on a maximum of    30K grid points per group
      compared to an average of    30K points (weighted)

    Estimated parallel speedup is     4.0
```

Finally, the actual parallel performance is written to standard out at the end of the run as shown below.

**Example 3.22**

```
GROUP TIMING SUMMARY   (Time each group spent in OVERFL)
 (*) STEP loop, (/) Chimera BC, (a) Adapt, (D) DCFCRT, (s) Grid & Q save

            0          25           50           75          100
            |-----------|------------|------------|-----------|
Group:   1  |************************///aaaaaaaaaaaaaaaaaaaDDs    97%
Group:   2  |************************///aaaaaaaaaaaaaaaaaaaDD     98%
Group:   3  |************************///aaaaaaaaaaaaaaaaaaaDD     97%
Group:   4  |************************///aaaaaaaaaaaaaaaaaaaDDs    99%

 Overall Measured Parallel Efficiency:  97.9%
```

## 3.10 Test Cases

A number of example problems are included in the **test** subdirectory. These cases provide examples of both input and code execution. These test cases include

- Simple 2D cases (steady flow, single grid):
  - flat_plate, flat_plate_high_re
  - flat_plate_wf (wall-function test case)
  - shear_layer
  - driven_cavity_2d
  - curved_wall_2d (curvature correction test case)
  - 3gas (simple multiple species convection case)
  - nozzle (rocket nozzle inflow/outflow boundary conditions)
  - cylinder (inviscid Mach 8 flow), cyl_holden (viscous Mach 16 flow)
- Transonic 2D or axisymmetric cases (steady flow, single grid):
  - bump (axisymmetric bump, shock-induced separation)
  - naca, naca4412, naca_ogrid
  - et_axi, srb_axi
  - normal_jet_2d (2-gas, time-accurate jet-in-crossflow)
- 2D multiple grids:
  - af3_96 (multi-element airfoil)
  - cascade
  - eggers, seiner (jet plume test cases)
  - powered_nacelle (jet engine inflow/outflow boundary conditions)
  - airfoil_adapt (off-body solution adaption)
  - normal_jet_adapt (2-gas, time-accurate with off-body solution adaption)
- 2D moving body cases:
  - airfoil_drop_2d
  - rotating_paddle_2d
  - pitching_airfoil_2d
- Classical time-accurate cases:
  - shock_tube
  - vortex_convection, vortex_convection_HiO, lambVortex_convection
  - stokes_1st_problem (impulsively started plate)
  - oscillating_sphere (acoustic test case)
- Subsonic/transonic 3D (steady, single grid):
  - m2129_s_duct (S-duct inlet)
  - rotating_disk (infinite rotating plate)
  - onera_m6 (classic transonic wing test case)
  - inf_swept (infinite swept wing)
  - ogive_cylinder
- Subsonic/transonic 3D (steady, multiple grid):
  - wingbody (AGARD test case)

- bizjet (assembling and running a wing/body/pylon/nacelle)
- robin_sym (helicopter fuselage, illustrates some numerical problems)

# References


1. Suhs, N.E., Rogers, S.E., and Dietz, W.E.,"PEGASUS 5: An Automated Pre-Processor for Overset-Grid CFD," AIAA-2002-3186, June 2002.
2. Noack, R., "SUGGAR: A General Capability for Moving Body Overset Grid Assembly," AIAA-2005-5118, Jun. 2005.
3. Chan, W.M., and Buning, P.G., "User's Manual for FOMOCO Utilities – Force and Moment Computation Tools for Overset Grids," NASA TM 110408.
4. Boger, D., and Dreyer, J., "Prediction of Hydrodynamic Forces and Moments for Underwater Vehicles Using Overset Grids," AIAA-2006-1148, Jan., 2006.
5. Pandya, S.A., Venkateswaran, S., and Pulliam, T.H., "Implementation of Preconditioned Dual-Time Procedures in OVERFLOW", AIAA-2003-0072, Jan. 2003.
6. Potsdam, M.A., Sankaran, V., and Pandya, S.A., "Unsteady Low Mach Preconditioning with Application to Rotorcraft Flows," AIAA-2007-4473, June 2007.
7. Baldwin, B.S., and Lomax, H., "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA-78-0257, Jan. 1978.
8. Baldwin, B.S., and Barth, T.J., "A One-Equation Turbulence Transport Model for High Reynolds Number Wall-Bounded Flows," AIAA-91-0610, Jan. 1991.
9. Spalart, P.R., and Allmaras, S.R., "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA-92-0439, Jan. 1992.
10. Wilcox, D.C., "Reassessment of the Scale-Determining Equation for advanced Turbulence Models, *AIAA Journal*, Vol. 26, No. 11, 1988, pp. 1299-1310.
11. Menter, F.R., and Rumsey, C.L., "Assessment of Two-Equation Turbulence Models for Transonic Flows," AIAA-94-2343, June 1994.
12. Degani, D., and Schiff, L.B., "Computation of Supersonic Viscous Flows Around Pointed Bodies at Large Incidence," AIAA-83-0034, Jan. 1983.
13. Nichols, R.H., "Algorithm and Turbulence Model Requirements for Simulating Vortical Flows," AIAA-2008-0337, Jan. 2008.
14. Suzen, Y.B., and Hoffmann, K.A., "Investigation of Supersonic Jet Exhaust Flow by One- and Two-Equation Turbulence Models," AIAA-98-0322, Jan. 1998.
15. Abdol-Hamid, K., Pao, S., Massey, S., and Elmiligui, A., "Temperature Corrected Turbulence Model for High Temperature Jet Flow," AIAA-2003-4070, June 2003.
16. Spalart, P.R., Deck, S., Shur, M.L., Squires, K.D., Strelets, M.K., and Travin, A., "A New Version of Detached-Eddy Simulation, Resistant to Ambiguous Grid Densities," *Theor. Comput. Fluid Dyn.*, Vol. 20, 2006, pp. 181-195.
17. Nichols, R., "Comparison of Hybrid RANS/LES Turbulence Models for a Circular Cylinder and a Cavity," *AIAA Journal*, Vol. 44, No. 6, June 2006, pp. 1207-1219.
18. Sickles, W.L., and Steinle, F.W., Jr., "Global Wall Interference Correction and Control for the NTWC Transonic Test Section," AIAA-97-0095, Jan. 1997.